Aufgabe 1

Wenn man für die α-Reduktion $\lambda x.t \underset{\alpha}{\rightarrow} \lambda y.\$^x_y\ t$ auf die Bedingung $y \notin \mathtt{Var}(t)$ verzichtet, kann eine solche Reduktion die Semantik verändern. Geben Sie dafür ein Beispiel an.

Aufgabe 2

Wenn man für die β-Reduktion

$$(\lambda x.t)s \underset{\beta}{\rightarrow} \$^x_s t$$

auf die Forderung $\mathtt{Fr}(s) \cap \mathrm{Geb}(t) = \varnothing$ verzichtet, kann eine solche Reduktion die Semantik verändern. Geben Sie dafür ein Beispiel an.

Aufgabe 3

Konstruieren Sie einen λ-Ausdruck t, der keine Normalform besitzt und dessen Reduktion zu immer größeren Ausdrücken führt.
Idee:

$$\lambda x.x(x)\lambda x.x(x) : D \in A_\lambda für\ alle\ x \in x^D$$

Sollte eigentlich dazu führen, dass x sich immer wieder selbst aufruft und damit keine Normalform möglich ist.

Idee Hinnerk: Für das ungetypte Lambdakalkül müsste folgendes funktionieren:

$$\begin{aligned} &(\lambda x.xxx)(\lambda x.xxx) \\ \Rightarrow &(\lambda x.xxx)(\lambda x.xxx)(\lambda x.xxx) \end{aligned}$$

So sollte auch die Bedingung erfüllt sein, dass die Reduktion den Ausdruck größer macht.
Ich weiß allerdings nicht genau, wie das in getyptem Lambdakalkül aussehen müsste (wobei das ja auch nicht explizit gefragt ist).. ;)

Aufgabe 4

Schreiben Sie je einen getypten λ-Ausdruck für folgende Aufgaben:

a) Eine symmetrische Funktion soll dreifach auf ein Argument angewendet werden.

$$\lambda fx.f(f(f(x))) \qquad\qquad [D \rightarrow D] \rightarrow D \rightarrow D$$

b) Gegeben sei eine Liste der Länge 4 von Elementen des Typs D und eine Funktion vom Typ $[\mathtt{D} \rightarrow \mathtt{D}]$, berechne die Anwendung dieser Funktion auf alle Listenelemente.

$$\lambda L\ g. < g(\pi_1 L); g(\pi_2 L); g(\pi_3 L); g(\pi_4 L) >$$

c) Beschreibe den uncurry-Operator im getypten λ-Kalkül, der angewendet auf eine Funktion vom Typ $[\mathtt{D}_1 \rightarrow [\mathtt{D}_2 \rightarrow \mathtt{D}_3]]$ eine Funktion des Typs $[(\mathtt{D}_1 \times \mathtt{D}_2) \rightarrow \mathtt{D}_3]$ liefert, wobei für alle `f`, a und b

```
(uncurry f) <a,b> = f a b
```

gelten soll.